ANL/CPS-20/2

Enhanced Visualization Capabilities in OpenMC: Report of Activities under a DOE GAIN Voucher

Computational Science Division

About Argonne National Laboratory
Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Lemont, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free at OSTI.GOV (http://www.osti.gov/), a service of the U.S. Dept. of Energy's Office of Scientific and Technical Information

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703)
605-6000 Fax: (703) 605-6900
Email: **orders@ntis.gov**

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728
Email: **reports@osti.gov**

Disclaimer
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

ANL/CPS-20/2

Enhanced Visualization Capabilities in OpenMC: Report of Activities under a DOE GAIN Voucher

prepared by
Patrick C. Shriwise and Paul K. Romano

Computational Science Division

September 22, 2020

1 Project Summary

This work was performed under a Cooperative Research and Development Agreement (CRADA) through the U.S. Department of Energy's Gateway for Accelerated Innovations in Nuclear (GAIN) program in partnership with Oklo Inc., the industry participant. The broad goal of this project was to improve the visualization capabilities of OpenMC related to model development and simulation results. OpenMC [?] is a Monte Carlo particle transport code that is developed and maintained by a broad community of contributors, of which Argonne National Laboratory plays a leading role.

2 Objectives and Requirements

The aim of this project was to create a stand-alone plotting application for OpenMC to aid in model development and post-simulation visualization of the spatial distribution of tally results. Specific requirements of the application were agreed upon at the beginning of the project in a meeting with Oklo staff. These requirements included the following:

- In-memory image generation

  The OpenMC executable can generate images of OpenMC models when run in "plotting mode" using plot parameters that are specified in an XML file format. In this mode, image files are written to disk in the PPM format. Images will be generated using calls to the OpenMC shared library so that image generation can occur without repeatedly invoking the OpenMC executable.

- Interactive plot image generation

  Updates to the displayed image in the application will be "live," enabling the user to explore the model with little to no delay between views.

- Model visualization

  Various model properties will be targeted for visualization during model preparation and verification. Users will be able to select images colored by cell IDs and material assignments as well as views of the temperature and density properties of these domains.

- Data visualization

  Data generated in OpenMC simulations is to be viewable in the application. OpenMC tally information will be viewable simultaneously with the model's geometric information. Visible tally information will be selectable by tally filter bin, nuclide, and score.

- Output data formats

  The application will be able to export to a variety of standard 2D image formats for use in sharing model information as well as figure generation for publication. Support for data export of both model information and tally data to 3D formats will be explored.

3 Work Performed

The following subsections describe work performed under the CRADA to develop an application for visualization that meets the above requirements.

3.1 In-memory Image Generation

An initial version of the plotting application written in Python 3 was developed that called the OpenMC executable in geometry plotting mode to write an image to disk that would then be read and displayed in the utility. After this initial demonstration was complete, changes were made so that the application could call functions from the OpenMC shared library in the same process without performing any disk I/O. To achieve this capability required expanding OpenMC's C/C++ API. The capability to generate two-dimensional data maps of cell and material IDs was exposed through OpenMC's C/C++ API, which is accessible to the plotting application by using Python bindings to the C/C++ API. These ID maps are used to create images in the plotter, colored by the property and ID at each location. This method for in-memory image generation greatly increased the interactive capability of the application by removing the need for initialization of the model and writing/reading the image from disk for each image displayed. Next, the ability to generate in-memory arrays of density and temperature was added to OpenMC and exposed through the C/C++ API in a similar manner. Support was then included for plotting of density and temperature with customizable colormaps, data limits, and transparency. Additionally, several plotting performance improvements were added in OpenMC, allowing for increased image resolution without loss of interactivity in the application. To complete in-memory generation of model images, the ability to evaluate surface, cell, and universe bounding boxes was added to OpenMC's C/C++ API; these bounding boxes are used to determine model bounds and generate default views in the application. With these changes, the application can generate images completely based on an in-memory instance of OpenMC.

3.2 Interactive Image Generation

The use of an existing OpenMC instance and the C/C++ API to generate model images greatly reduces the time needed to generate images in the application, providing a better user experience. Without the need to reinitialize OpenMC, generation of the cell/material ID maps and density or temperature maps can begin as soon as a new view of the model is requested. This capability is particularly important for visualizing densities because generating a density map requires the simulation cross sections to be loaded during initialization, which would have made the time necessary to generate these images prohibitive in the application's original design.

While adding density and temperature visualization using the C/C++ API, the underlying code for plotting in OpenMC was also refactored by generalizing the generation of material/cell IDs and densities/temperatures to improve performance of image generation by 2–3×. This improvement enables interactive image generation at a higher resolution, providing a better user experience, and also applies to image generation performed by using the OpenMC executable run in plotting mode. The application also fully leverages OpenMC's shared-memory parallelism through OpenMP, giving users a means for further reducing the time needed to generate images for publication figures or presentations. Control of image resolution provides control over the speed of image generation as well.

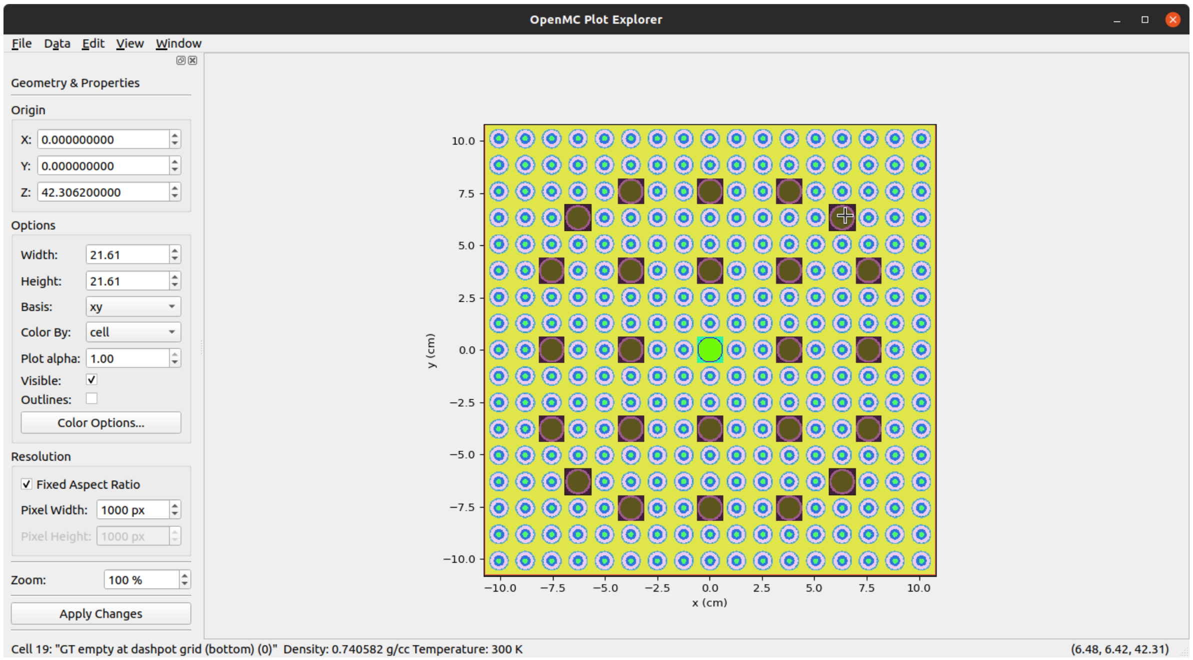

Figure 1: The plotting application and geometry dock (left side) displaying a cell-based image of a reactor assembly model.

3.3 Cells and Materials

OpenMC model views can be generated along any principal axis for either cell or material domains. A view's extents are specified by using an origin, width, and height in centimeters. These values are set by default when loading the application using the bounding box of the model's root universe. As shown in **??**, domain IDs, along with their density and temperature values, are displayed in the status bar and updated as the pointer moves and hovers over different domains. The coordinate values of the pointer's current location in centimeters are also displayed in the status bar in a live manner. Rather than updating the view, a user can adjust the zoom setting and pan to the area of interest if desired.

The color of each domain in the view is randomly generated initially, but each color can be customized either by using a context menu (shown in **??**) or in a color customization dialogue containing a color table for all cells/materials displayed.. The background color of the displayed image can also be customized if the view goes outside the model's bounds. Domains can also be added to a highlight and mask group, each with its own customizable color. These options, and others, are directly available via a context menu accessed by right-clicking on the domain of interest in the current image.

Figure 2: The plotter context menu accessed for a material in the reactor assembly model.

3.4 Model Properties

The refactor of OpenMC's plotting capability allows for image generation of density and temperature for given domains. OpenMC associates temperatures primarily with cells and density primarily with materials. These properties can be viewed by using a variety of different colormaps with customizable minimum and maximum values and linear or logarithmic scaling. The input temperature of different materials or cells is often a small set of fixed values, resulting in little additional information when viewing temperatures as opposed to materials. However, **??** shows a temperature plot of a pincell model from a multiphysics calculation using OpenMC, where distributed cell temperatures may vary independently. The plotting application provides a fast method for viewing and verifying model properties of these types of results.

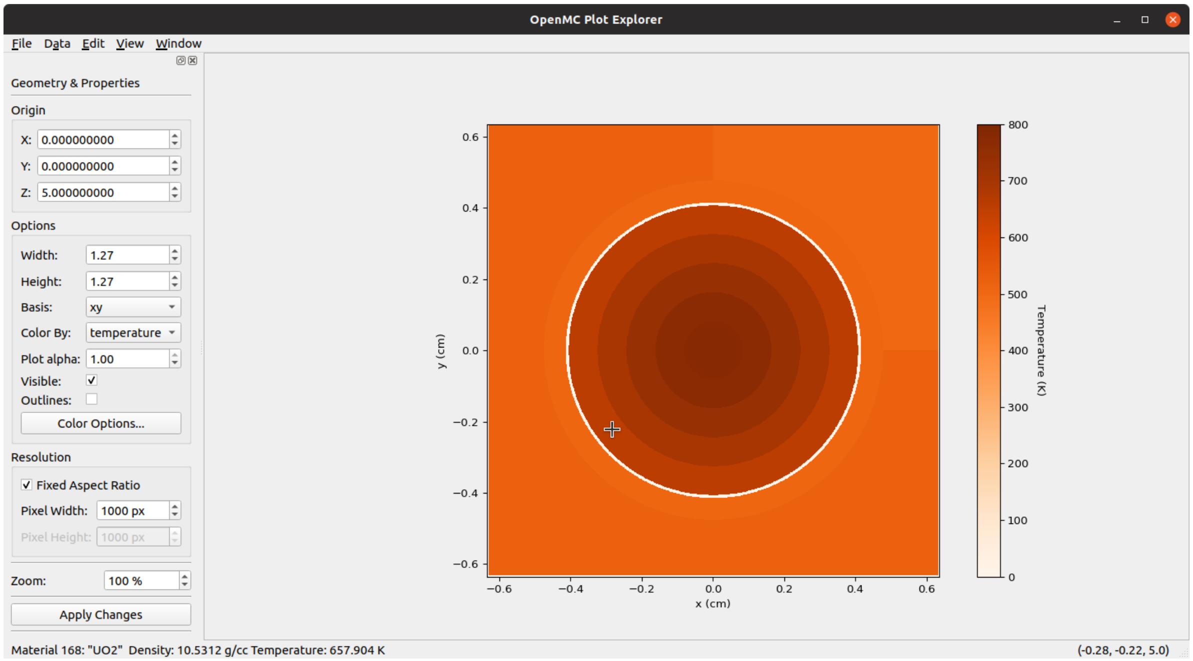

Figure 3: Temperature plot of a pincell model from a multiphysics calculation using OpenMC.

3.5 Geometry Errors

The ability to identify geometry overlaps was added to OpenMC's plotting capability. This check performs a point containment test for each pixel location on all cells in the model rather than stopping at the first cell found to contain that point. Locations found to be in more than one cell are classified as locations in an overlapping region and can be identified with a customizable color as well as an identifier in the application status bar. The model in **??** contains an overlapping region, shown in red. This feature is intended as a way for users to search for errors in their geometry. This feature is currently optional because it is time consuming to perform the additional point containment tests necessary to display this information.

3.6 Tally Data: Spatial Filters, Bins, Scores, and Nuclides

OpenMC's simulation results are written in an application-specific HDF5 format as statepoint files. These files can be loaded into the plotting application to view tally results containing spatial filters (cell, material, universe, or mesh filters). Since the filters for a tally have already been applied during the simulation, they cannot be added or removed from the tally data when loaded into the plotting application. The bins of these tallies, however, can be selected or unselected to view a subset of the data for each filter. Similarly, subsets of tally scores and nuclides can be selected in the application provided that their units can be combined meaningfully. Menus for selection of bins, scores, nuclides, and the visible tally value are all automatically populated based on the current tally selected. A distinction is made between tallies using a mesh filter and those without during tally image generation. Mesh filter bins represent each voxel of the mesh and are enabled by default when viewing a tally with a mesh filter. Other spatial filter bins can still be enabled/disabled when a mesh tally is present,

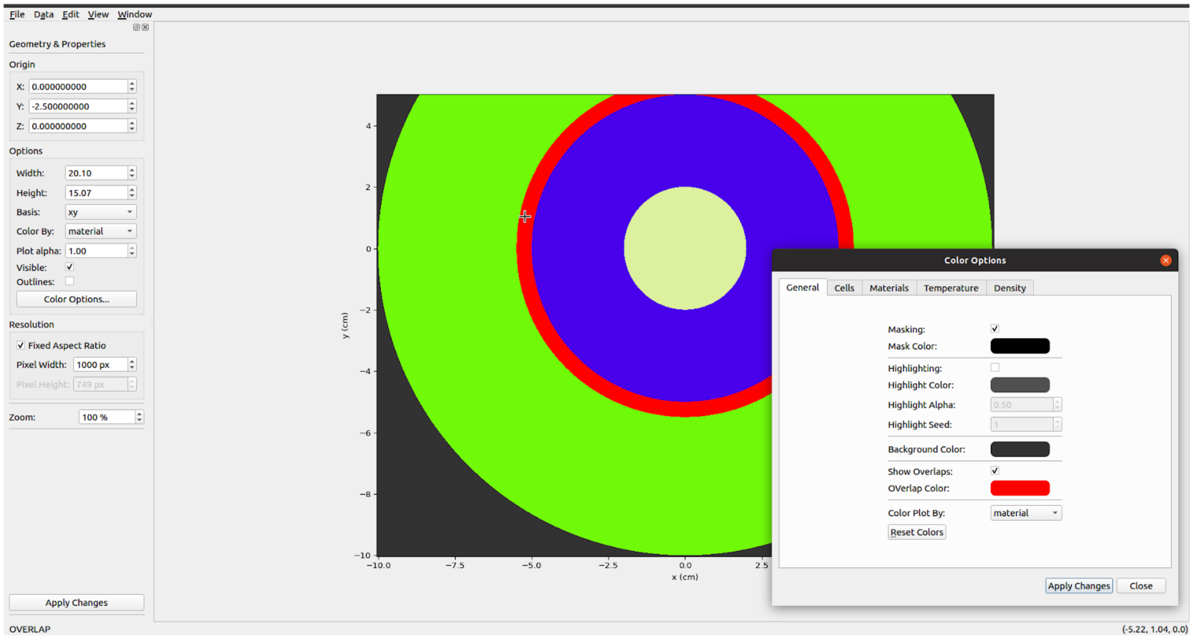

Figure 4: Display of an overlapping region in a pincell model (note the indicator in the status bar).

allowing the user to overlay mesh tally data on a specific domain in the view. **??** shows a tally with both mesh and cell filters applied. By selecting only the bins of cells one and three (the innermost and outermost regions of the pincell), the data corresponding to the middle ring, cell two, is removed from the image. The value of the currently displayed tally based on pointer location is also displayed in the application status bar, providing an exact value of the tally at a location of interest.

Tally data is currently managed through OpenMC's Python interface, separating it from the OpenMC instance connected to the application's geometry data. As such, statepoint files can be opened and closed in the same session to view results from several different simulations if desired. The statepoint data is expected to match the in-memory model initialized when the application is started.

3.7 Geometry Overlay

Tally data is displayed on top of the geometric data, which can make it difficult to distinguish the two datasets from each other, particularly when viewing cell- or material-based tallies. To mitigate this problem, options have been added to remove the geometry colors from the canvas and enable outlines of geometry domains on the plot. These enable the user to display information with an understanding of the effect of domains on the tally data. Model domains can be masked as well if parts of the view overlap with the tally data. Domain information is still provided in the application status bar based on the pointer's location to assist with the user's orientation in the model and to indicate what combination of spatial tally bins will provide the desired image in an area of interest.

Additionally, tally data can be viewed by using isometric contours rather than colormaps,

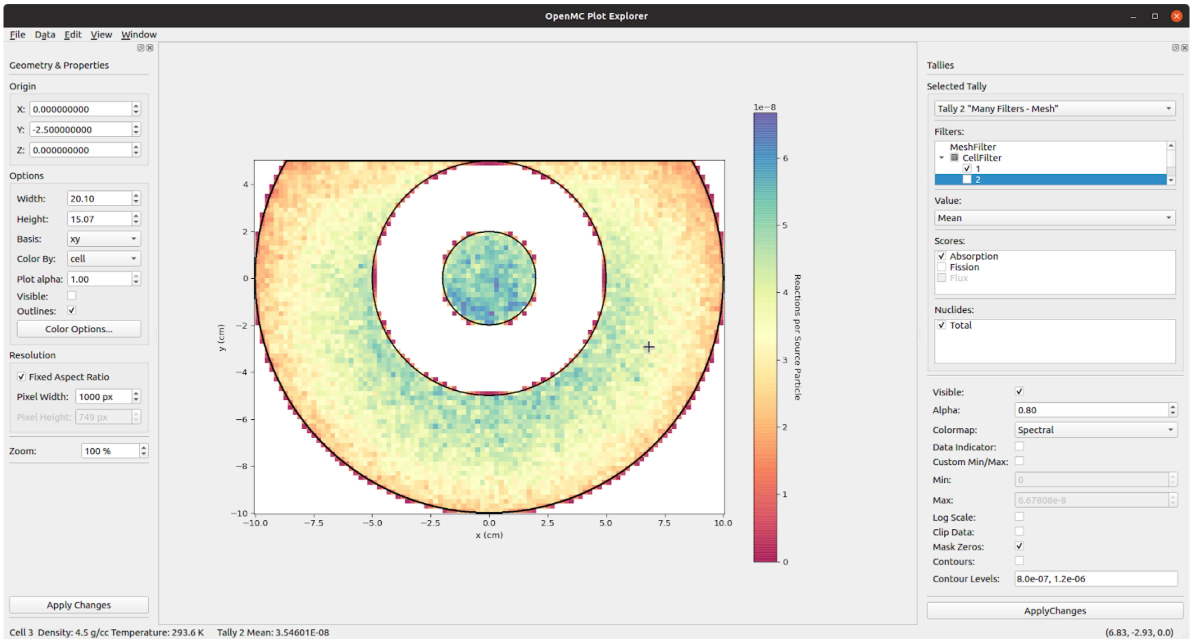

Figure 5: Image of mesh tally data for a pincell model with domain boundaries turned on. The tally values shown represent the absorption score only in cells one and three. The values in cell two are masked out by disabling the appropriate bin in the cell filter of the tally.

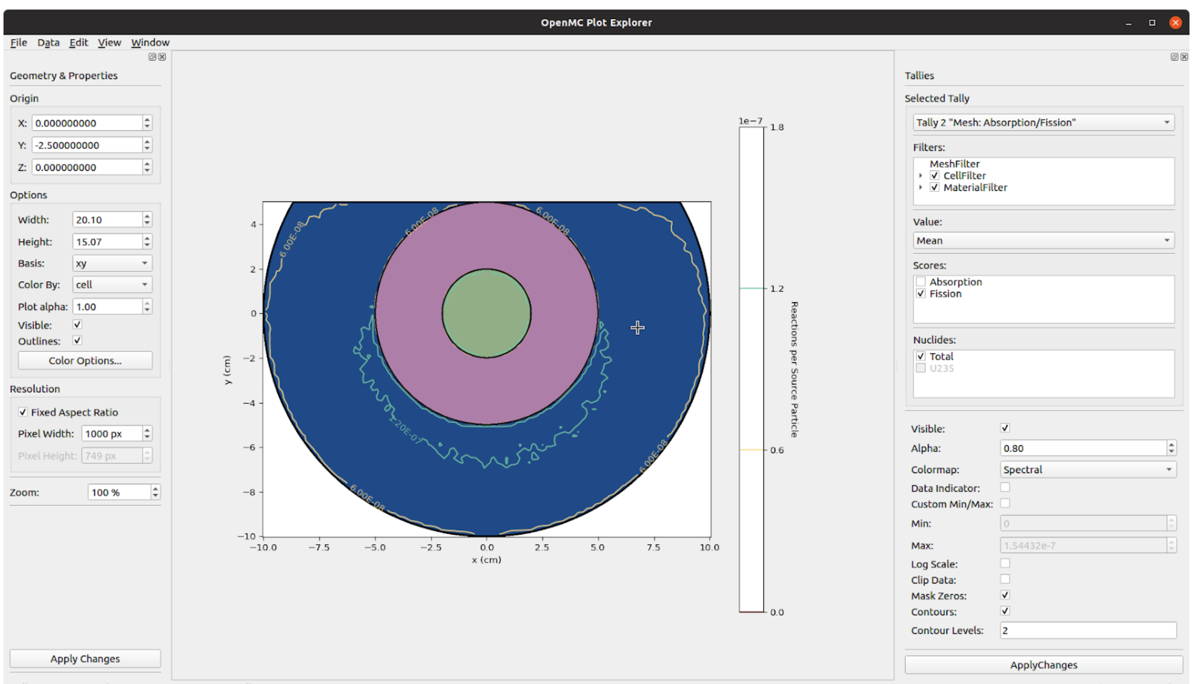

Figure 6: Contour plot of a mesh tally with the geometry displayed beneath.

as seen in **??**. The contour plot provides the user with an alternative method to interpret tally value gradients with respect to the geometric boundaries of the model while maintaining the colors of domain regions for visual context and orientation.

3.8 2D Image Formats

The plotting application leverages the Matplotlib Python library to display colormaps and contours representing geometric domains and tally data. With the use of this library, the application can export images in any format supported by the Matplotlib installation (.jpeg, .tiff, .png, .pdf, and others).

4 Additional Features

In addition to the capabilities described in **??** that meet the original requirements, several features were added to the plotting application that will further enhance the user experience. The following subsections briefly describe these features.

4.1 Undo/Redo

The application supports traversal of previous views with an "undo" capability. The user can also move forward through views with a "redo" capability. Together, the undo/redo capability enables users to return to a preferred view without having to memorize the specific settings of that view, which can be cumbersome to track if multiple settings have been altered between views.

4.2 Saved Views and Settings

Views can be saved from the application and loaded later by the user. Since the undo history in the application is limited, the ability to save views provides a way for users to recover specific views during a longer session. It also provides a way to share views with others. By providing model inputs and a saved view, another user can expect the same settings and image to be displayed. Each time the application is closed, the current plot settings are saved in a binary format to be reloaded the next time the application is opened for that model. This way, previous work like custom coloring of cells/materials is not lost between sessions.

4.3 Model Reloading

One of the purposes of the plotting application is geometry debugging. To facilitate this, the application allows the user to reload a model during a session so that changes to the input XML files can be viewed without restarting the application. This capability provides instant feedback for changes to the input XML files and is particularly useful when used side by side with a text editor during model preparation.

4.4 CAD-Based Geometry Visualization

OpenMC supports CAD-based geometry via the DAGMC toolkit [**?**], a software library from the Computational Nuclear Engineering Research Group at the University of Wisconsin–Madison. DAGMC enables particle tracking on tessellated CAD surfaces composed of triangles. OpenMC's geometric operations, including the point containment test used to produce images, are agnostic to whether the domain entities are formed by OpenMC's native Constructive Solid Geometry (CSG) representations or on DAGMC's tessellated CAD surfaces. As a result, the plotter works on an OpenMC model setup for a DAGMC geometry without modification, although image generation may take significantly longer. For example, the image in **??** was generated by using a DAGMC file with no modification to the plotting application's code.

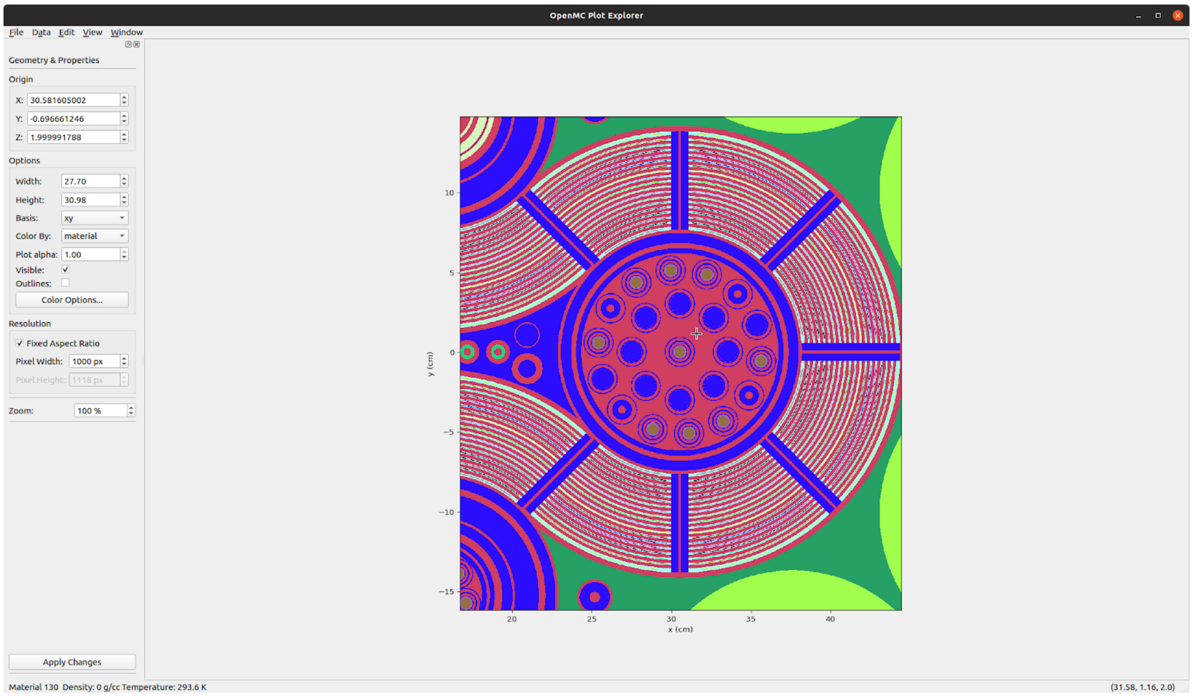

Figure 7: An image generated by using a CAD-based model of the Advanced Test Reactor, a difficult geometry to represent using CSG.

5 Application Roadmap

Given the usefulness of the plotting application developed under this CRADA to the wider OpenMC community, there is interest in continued support and development through other programs. In this section, we identify several areas that warrant further development.

5.1 Overlap Checking

Currently, overlaps are shown solely in the displayed view, which is useful only when the user has some idea of where the overlaps occur. A feature is planned that will search the model for overlaps. The user will be able to provide a resolution in all three dimensions over which the search is to occur. 2D maps of the domain will be generated at each level of this grid. If overlaps are found, the tool will halt and display that view on screen with overlap coloring enabled. If no overlaps are found, the tool will report this fact when the search is complete.

5.2 Documentation

Online, searchable documentation is to be provided for the application in the near future. This will include the plotter's data model, menu descriptions, features, shortcut listings, suggested workflows, and examples. Upon completion of comprehensive documentation, the application will be considered for incorporation into the main OpenMC GitHub repository to be shipped with the standard codebase.

5.3 3D Visualization

The intent for the plotting application is to enable 3D visualization by providing tools for exporting geometry and tally data to commonly supported formats for visualization in tools such as VisIt or Paraview. The OpenMC executable's "plotting mode" currently can produce an HDF5 voxel file containing a 3D Cartesian mesh where each mesh voxel is labeled by the cell ID, material ID, temperature, and/or density based on the point at the center of that voxel. A script is also provided with OpenMC that can translate these HDF5 files into the VTK format. In the short term, this method will continue to be used for generating 3D data of model properties. For tally data, a feature is in progress in the plotting application that will enable generation of 3D VTK files based on the current data selection for a tally. For tallies with mesh filters, the resulting VTK grid will match the mesh in the mesh filter. For tallies with other spatial filters, the user will be able to specify the bounds and resolution of the Cartesian mesh.

6 Conclusion

A plotting application has been developed to support the key features desired by both the OpenMC development team and members of Oklo Inc.'s design team as agreed upon at the beginning of the project. This application will be distributed with the main OpenMC codebase and will require only two additional Python third-party packages, both of which are in wide use and actively supported. The plotting application is already in use by members of the community to aid in model preparation, visualization, and validation. Our intent is for the OpenMC development team to continue supporting this project, adding features to the application beyond those discussed in the roadmap section above.

Acknowledgments

This material was based upon work supported by the U.S. Department of Energy, Office of Nuclear Energy through a Gateway for Accelerated Innovation in Nuclear (GAIN) voucher under Contract DE-AC02-06CH11357.

Computational Science Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Lemont, IL 60439

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC